HLA Radio Communication

Danny Cohen, Andreas Kemkes, Moshe Kirsh,

Ron Nagamati, Mike Robkin

PERCEPTRONICS

21010 Erwin Street, Woodland Hills, CA 91367

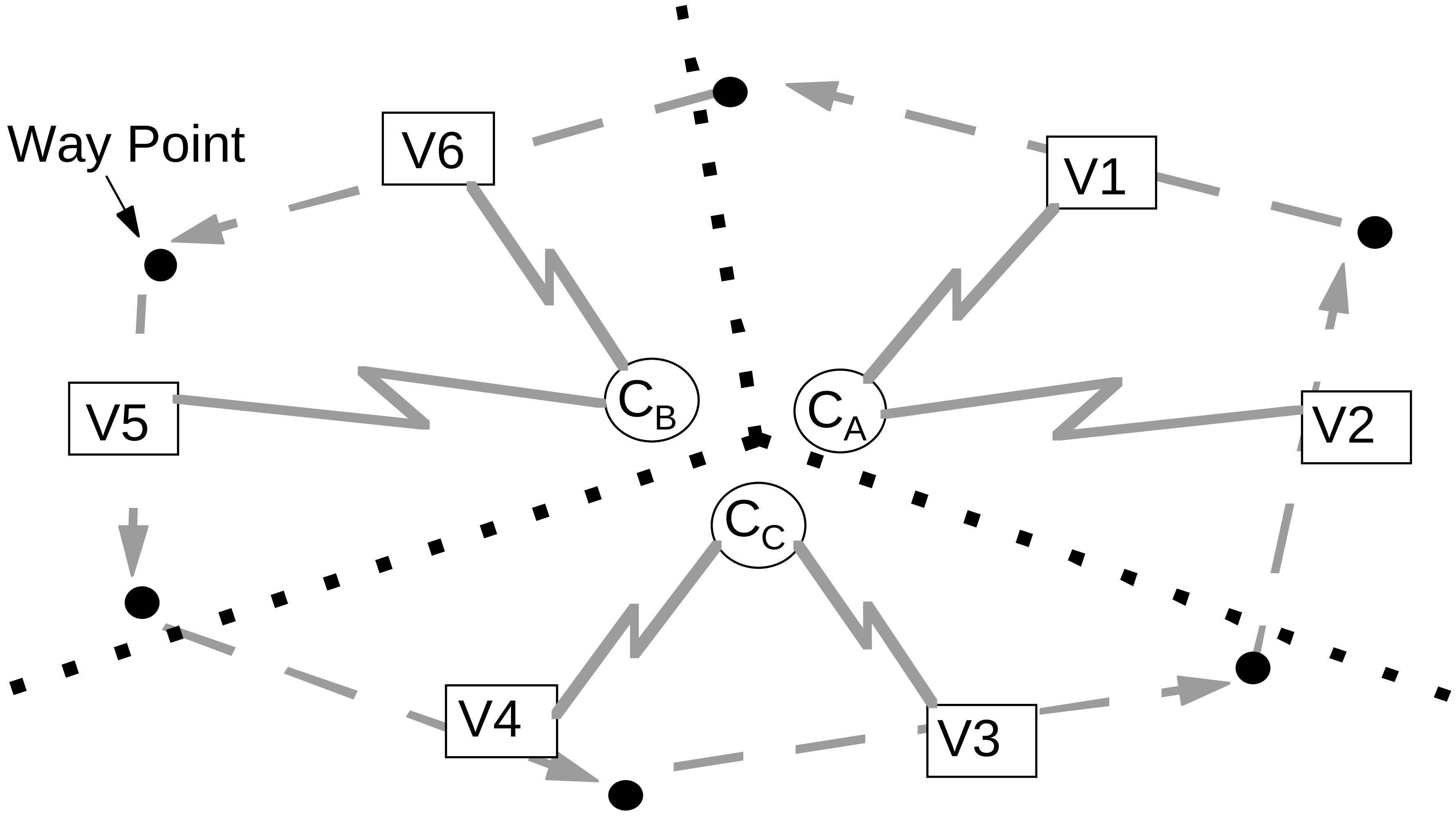
Acknowledgment

This work is funded by DMSO and STRICOM under contract No. N61339-95-C-0028.

Goals

- Demonstrate a pure HLA approach and promulgate approach, concepts, and lessons learned to the simulation community
- Help solve general communication problems
  - Use a simplified but realistic scenario
  - A communication simulation, not a radio simulation
- Expand to other domains (C4I & Air Traffic Control)

The Scenario: Communication Model



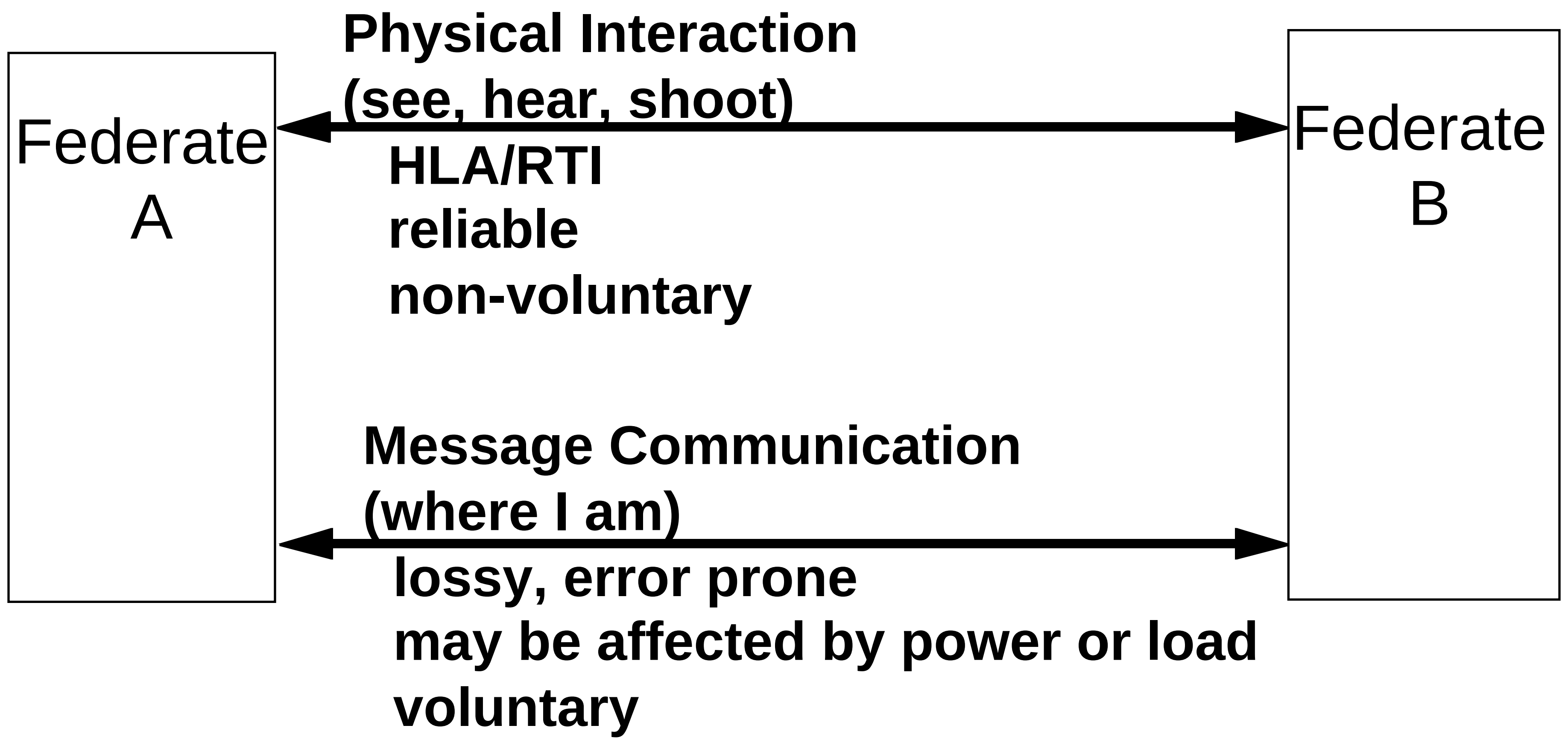
Vehicle Behavior

- Each Vehicle
  - listens to one commander at a time
  - is ordered to move to way points
  - is ordered to change frequencies
  - reports its position

Commander Behavior

- Each Commander
  - use his own frequency
  - controls a unique geographic area
  - orders vehicles to new way points in his area
  - orders vehicles to change frequencies

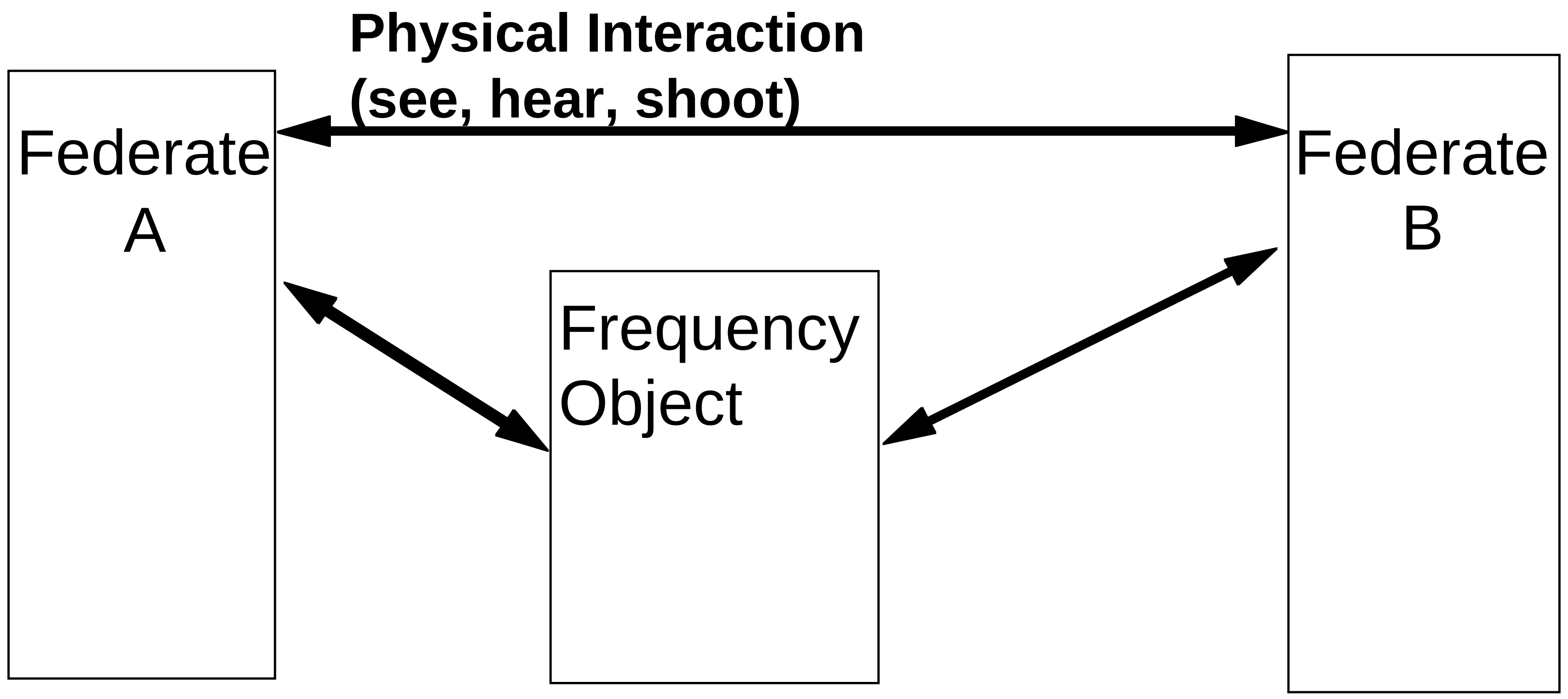
Communication Types



HLA Communication (1)

- The communication is implicitly handled by the RTI
- In order to simulate transmission problems (e.g. collision, interference, etc...) we introduce the FREQUENCY OBJECT

HLA Communication



System Design (1)

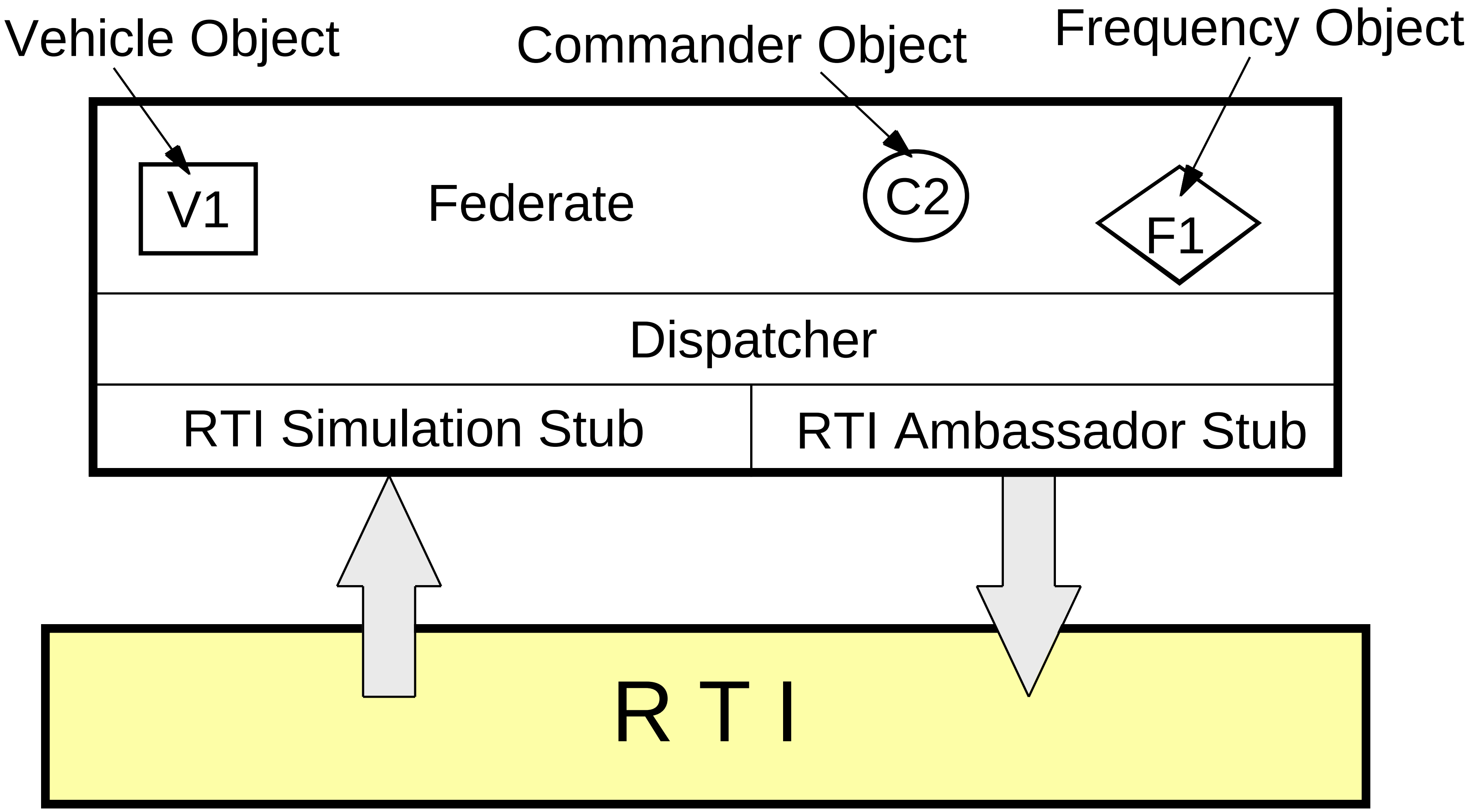
- Takes advantage of RTI features
- Logically centralized
- Physically distributed
- Scaleable
- Modular
- Supports filtering, jamming, & interference
- Allows for modeling and changing radios

System Design (2)

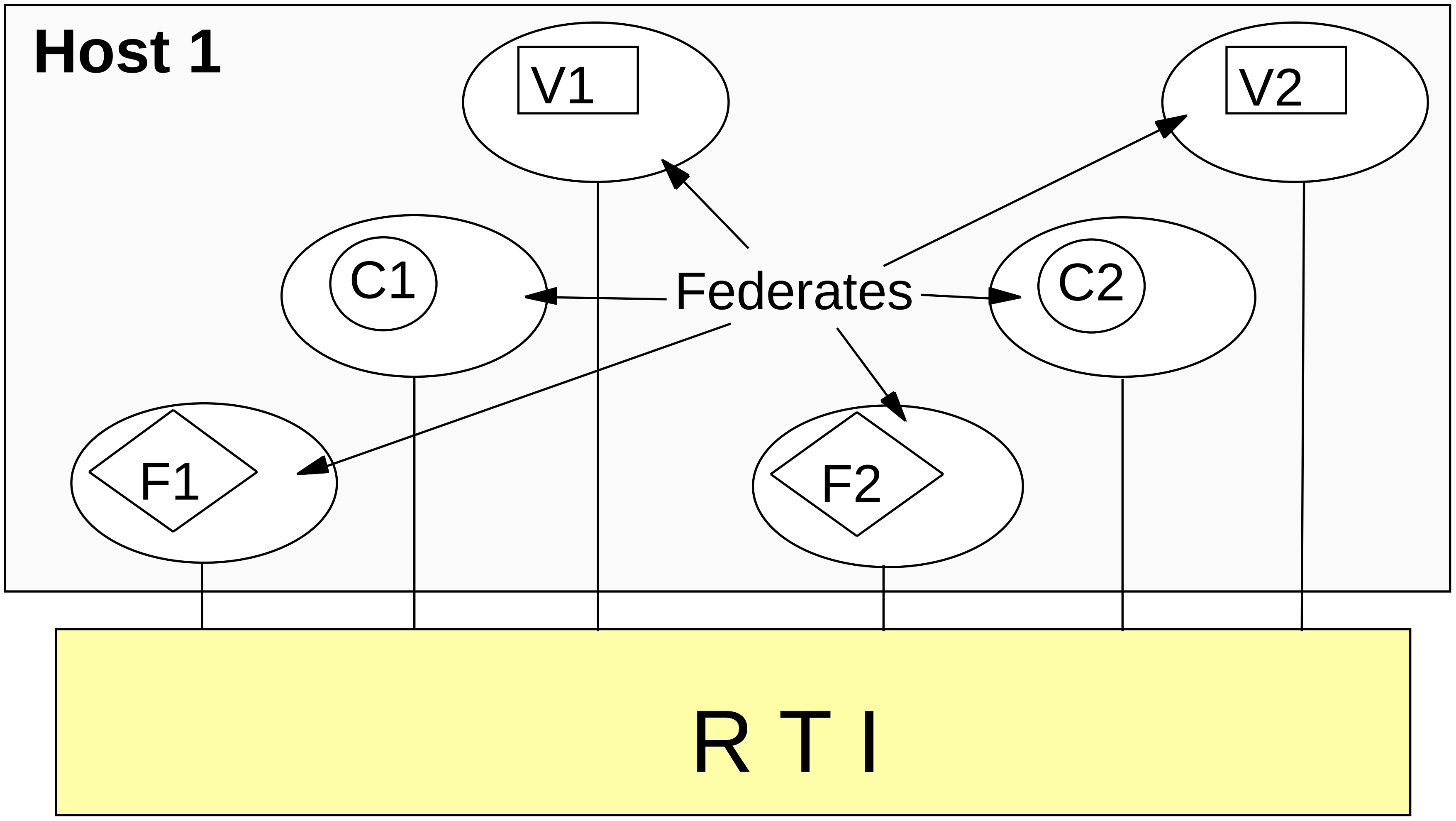
Distinguishes between:

- **Frequency Objects**
- **Radio Objects**
- **Vehicle Objects**
- **Commander Objects**

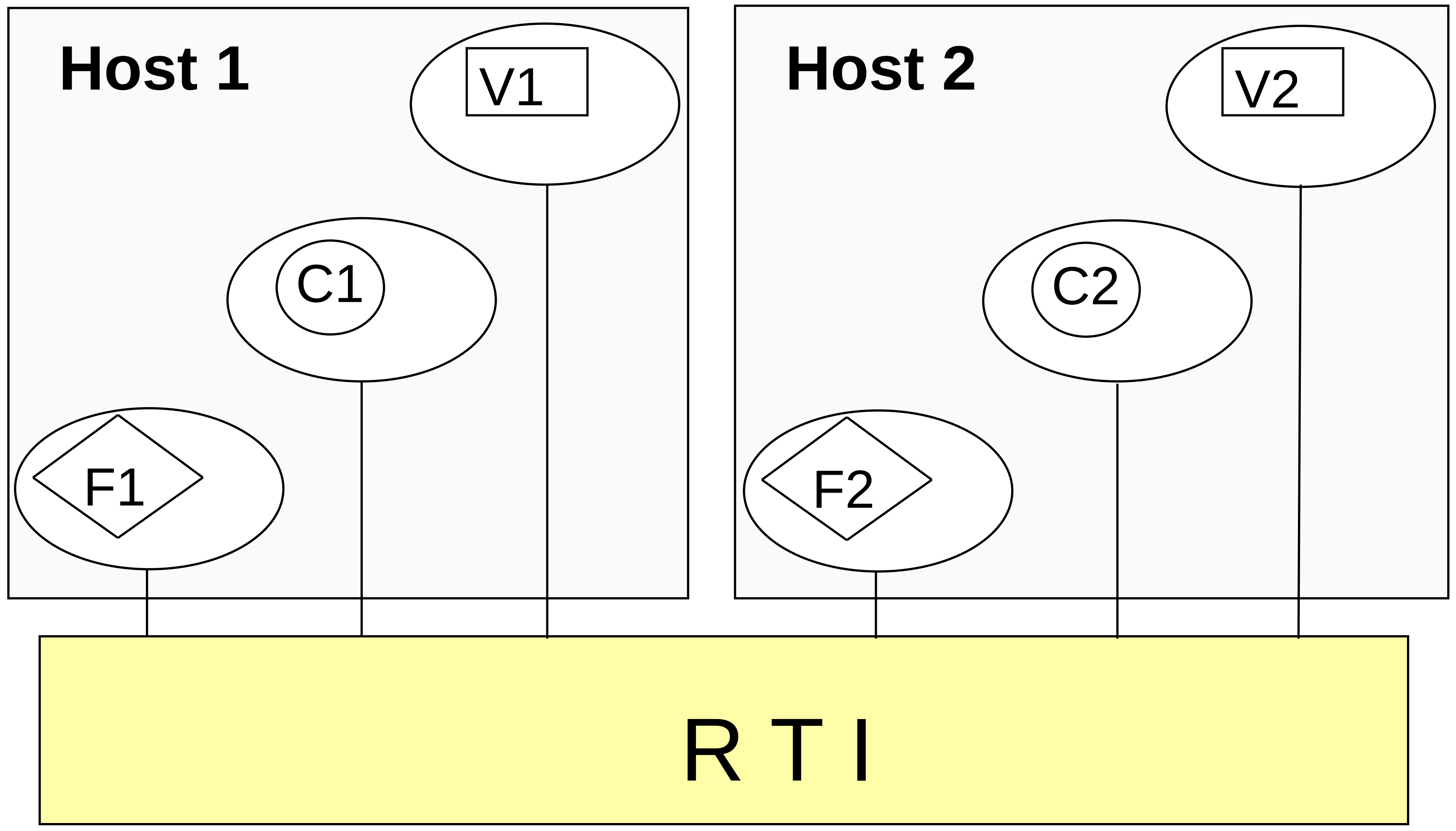
RTI Interface



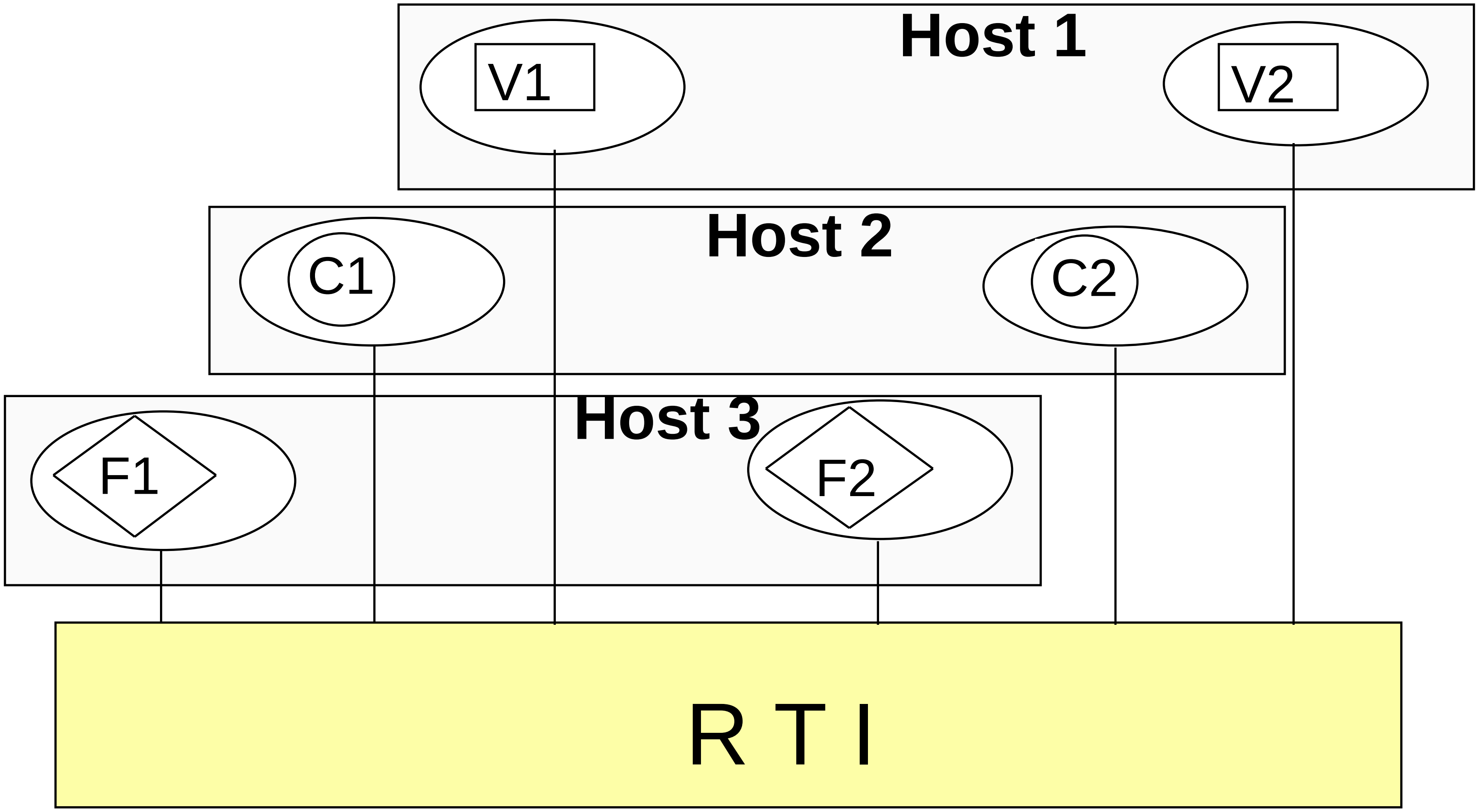
Configuration 1: Single Host



Configuration 2: Site or Operational Division



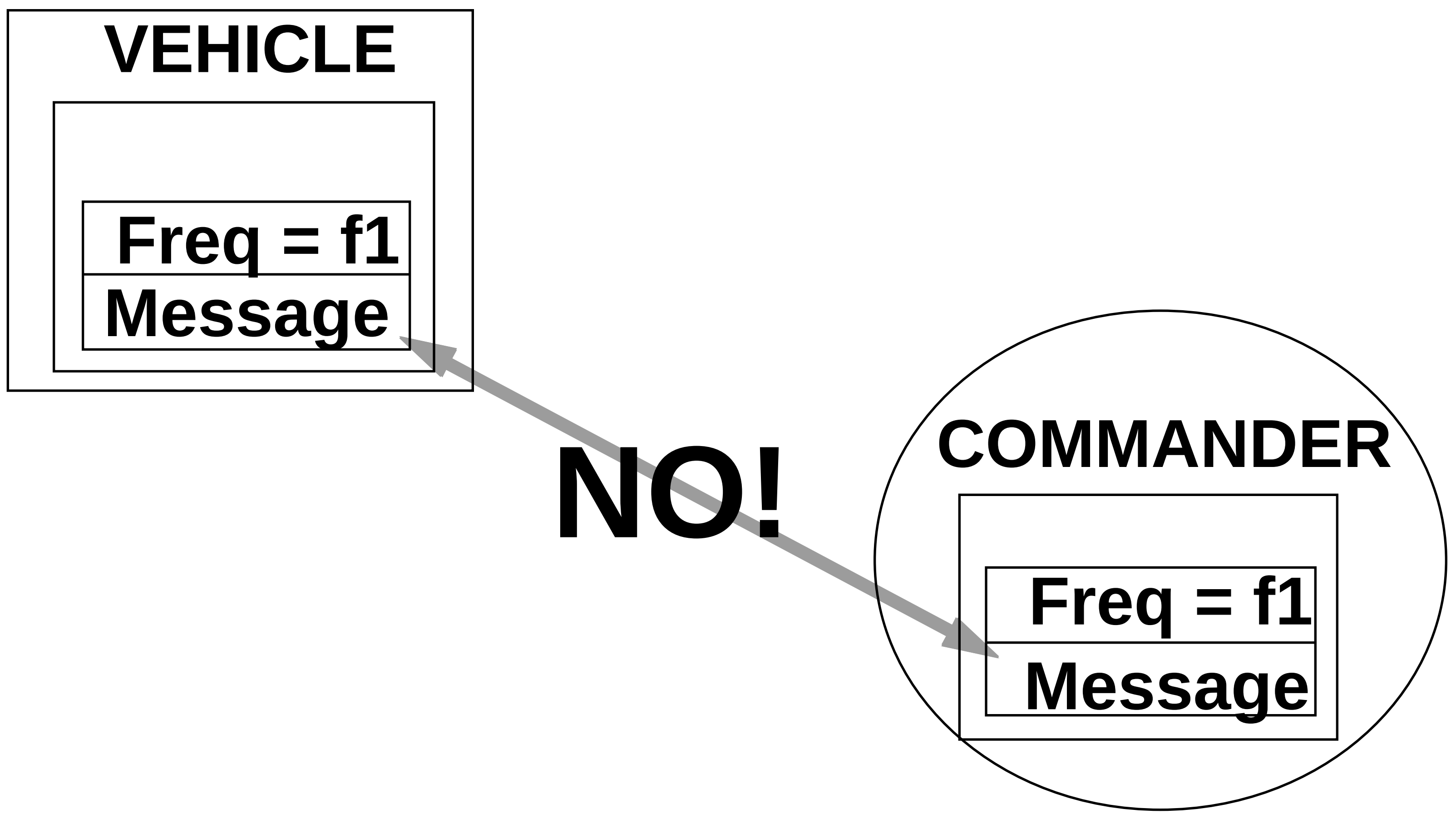
Configuration 3: Functional Processes



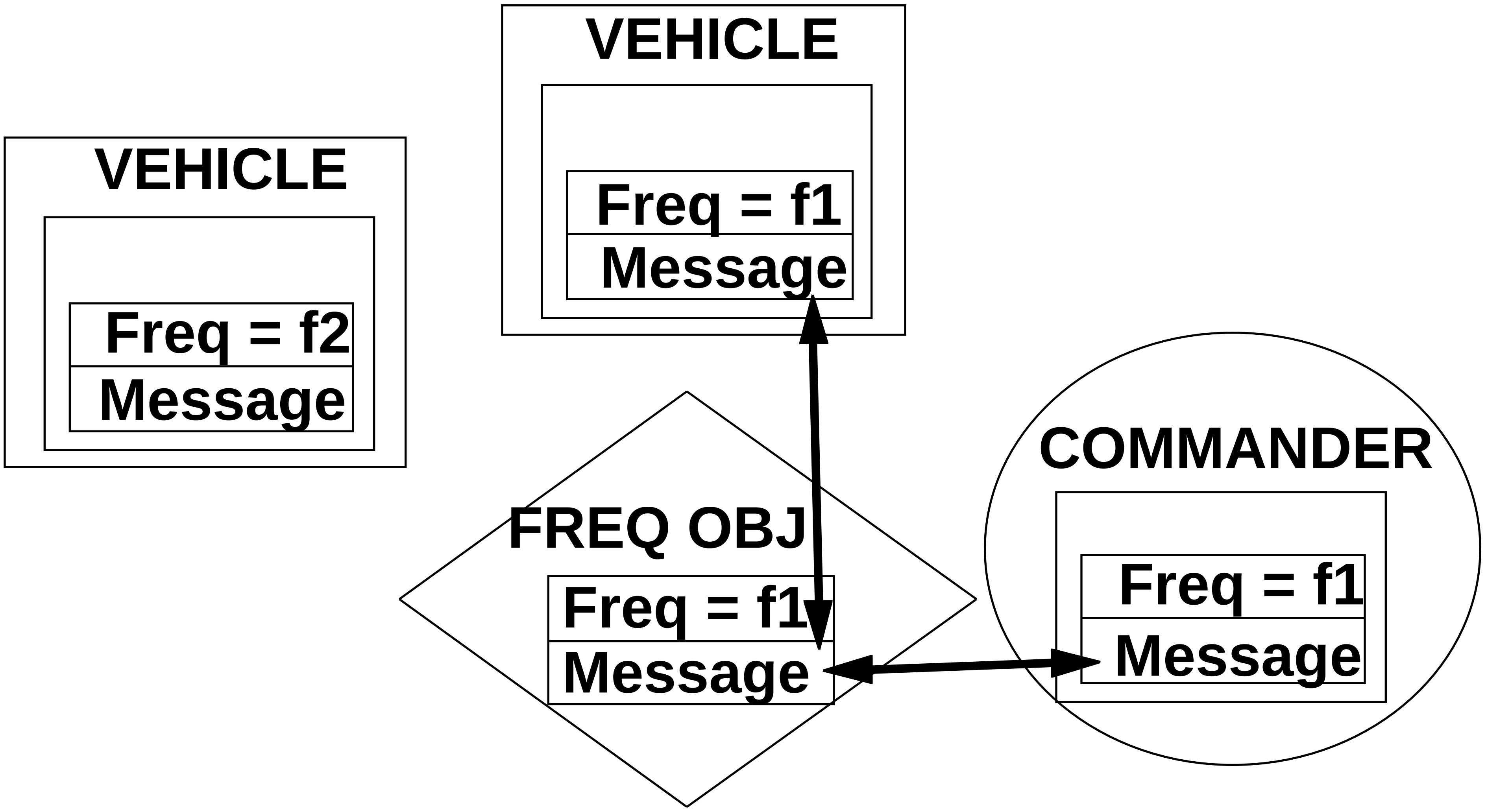
Software Design

- Message Passing
- Scenario Participants
- Frequency Behavior
- RTI Initialization Data (RID)
- Simulation Object Model (SOM)
- RTI Subscriptions
- Object Interaction
- Protocols

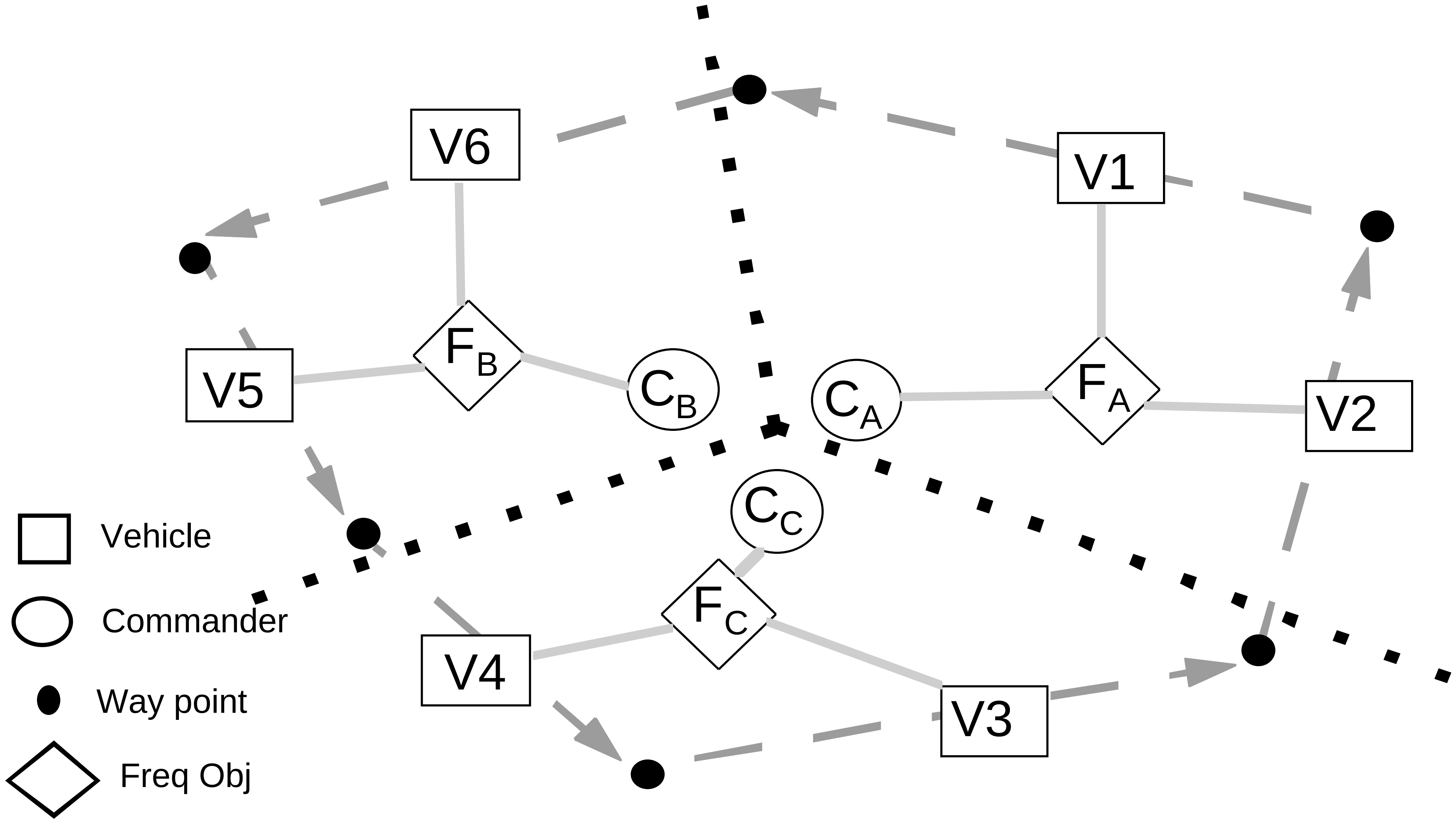
Message Passing (Wrong!)



Message Passing



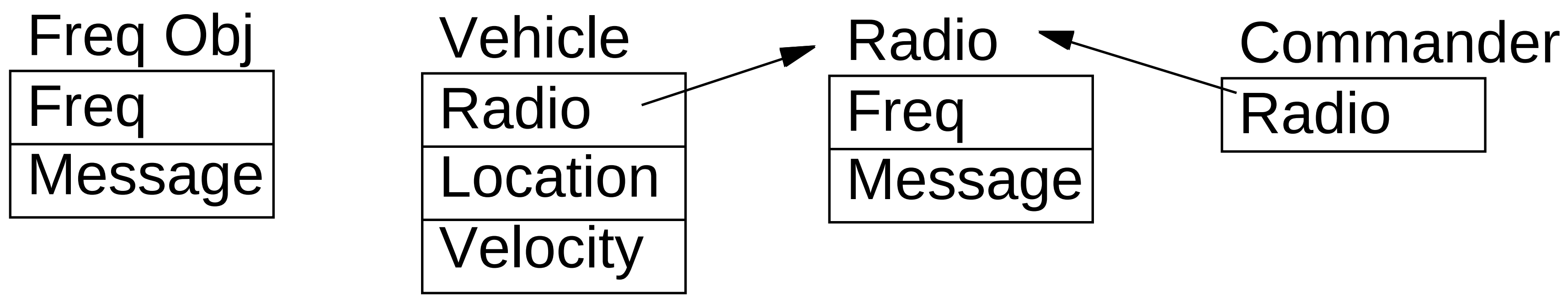
The Scenario



Participants

- RTI Objects
  - Radio, Frequency, Commander, Vehicle
- Federates
  - Vehicle, Commander, Frequency
  - Plan View Display

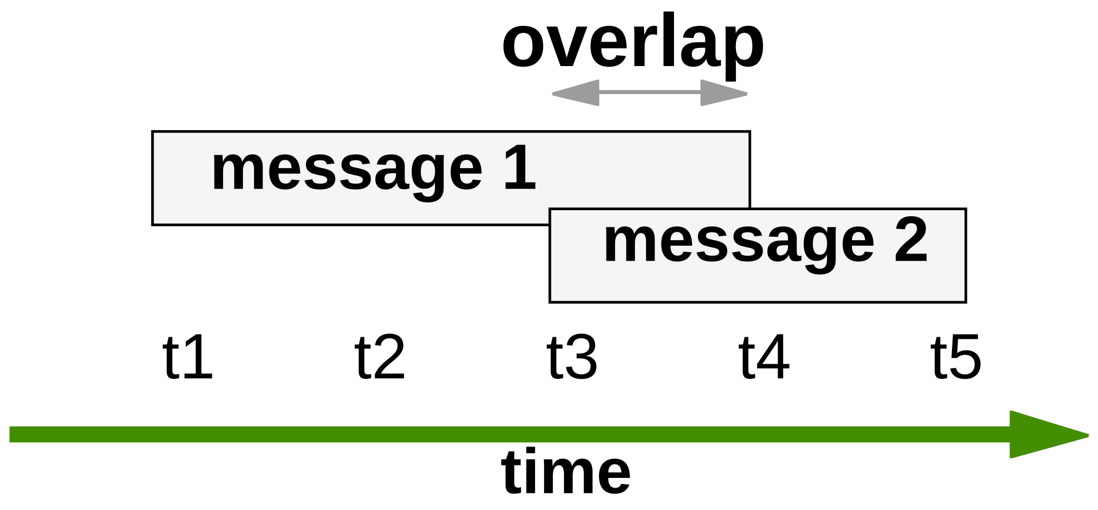
Classes and Attributes



Frequency Behavior

- Each Frequency Object
  - receives messages from objects on its frequency
  - detects and handles message collisions on its frequency
  - distributes the message to all objects on its frequency

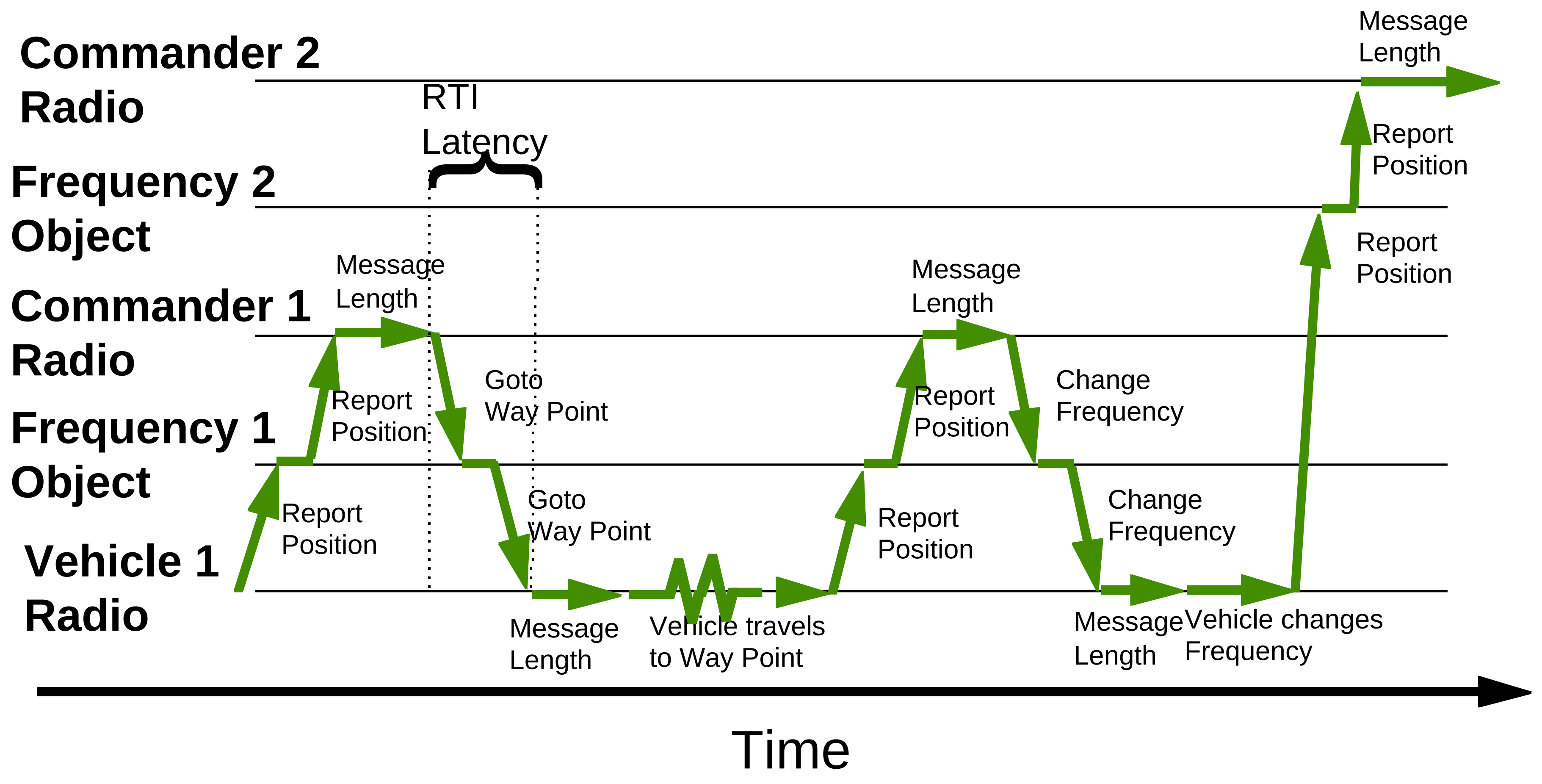
Message Collision



Collision could result in:

- •Losing message #2
- •Losing both messages
- •Degrading both messages

Event Time Line



RID
Radio Class

```
(objects
    (class radio
            (attribute frequency ...)
            (attribute message ...)  ;; complex attribute
            (attribute filterfrq ...) ;; for filtering ...)
```

RID
Frequency Class

```
(objects
    ...
    (class frequency
            (attribute frequency ...)
            (attribute message ...)  ;; complex attribute
            (attribute filterfrq ...) ;; for filtering ...)
```

RID Commander & Vehicle Classes

```
(objects
        ....
       (class commander
               (attribute location ...) ...)

       (class vehicle
               (attribute position ...) ...) )
```

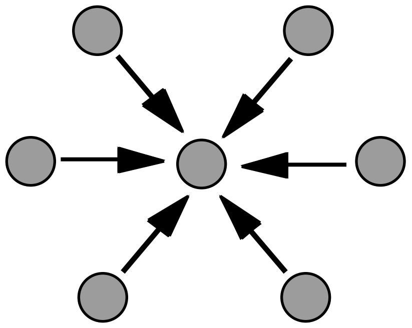
SOM of Frequencies

```
(objects
        (class radio
                (attribute frequency ...)
                (attribute message ...)  ;; complex attribute
                (attribute filterfrq ...) ;; for filtering ...)
        (class frequency
                (attribute frequency ...)
                (attribute message ...)  ;; complex attribute
                (attribute filterfrq ...) ;; for filtering ...)
)
```

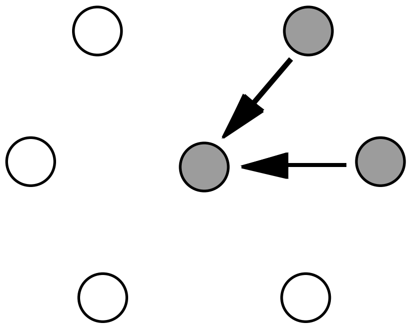
RTI Subscriptions

Subscription By Class (= Broadcast) ⇨ Specified attributes from **all instances** of a class will propagate to the subscribers

Subscription By ID (= Multicast) ⇨ Specified attributes from a **selected list of instances** of a class will propagate to the subscribers

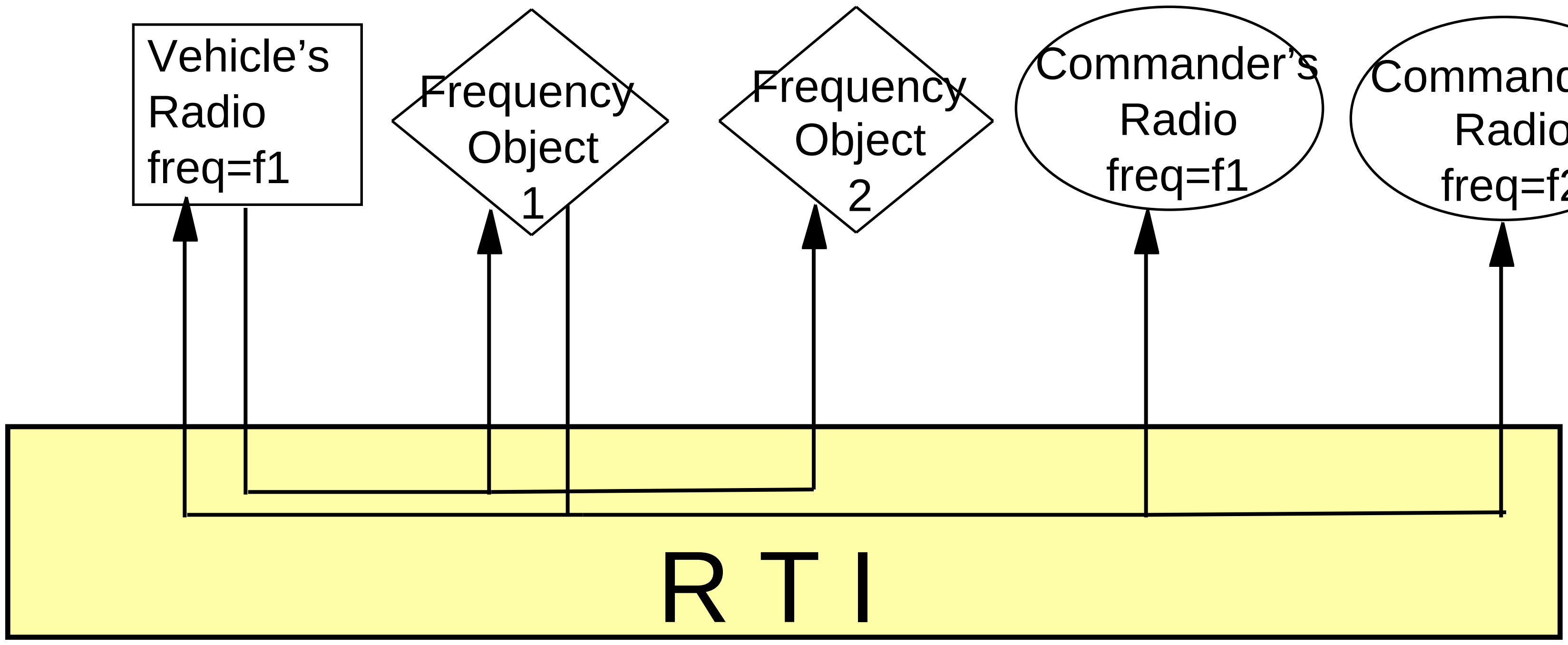


By Class

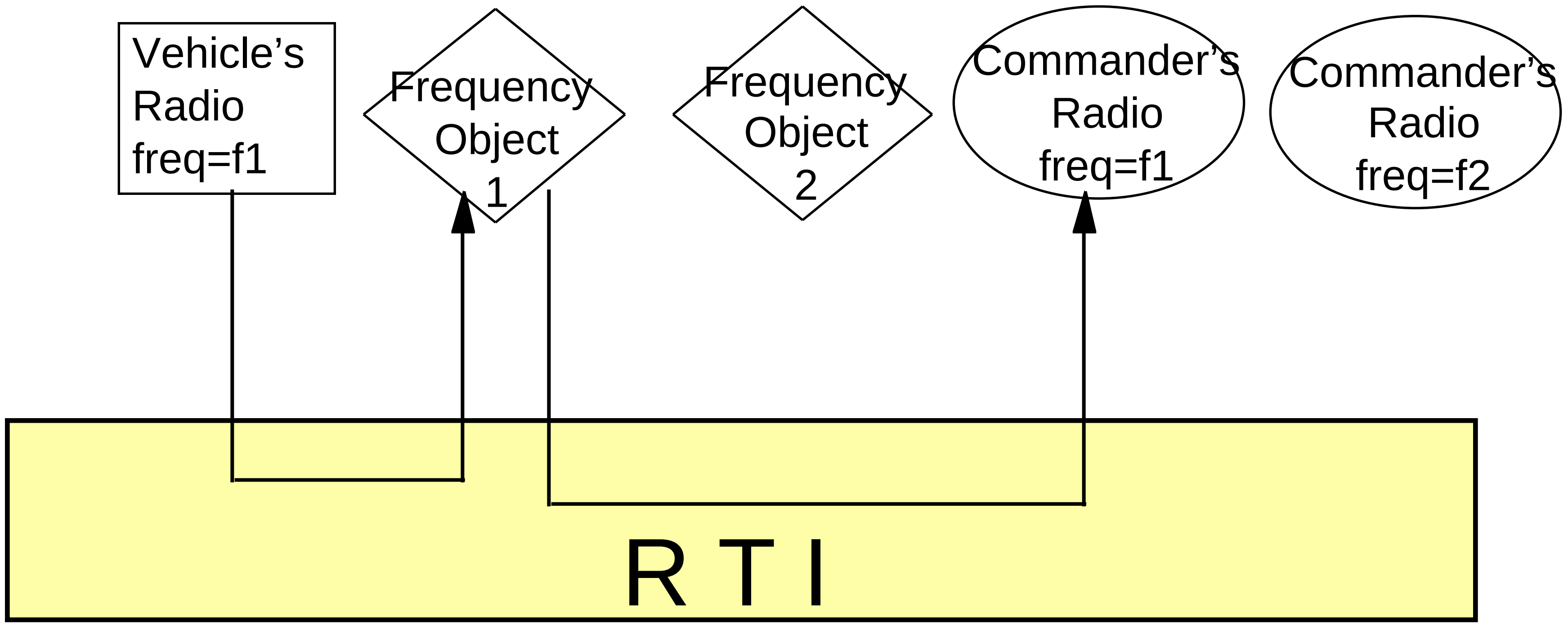


By ID

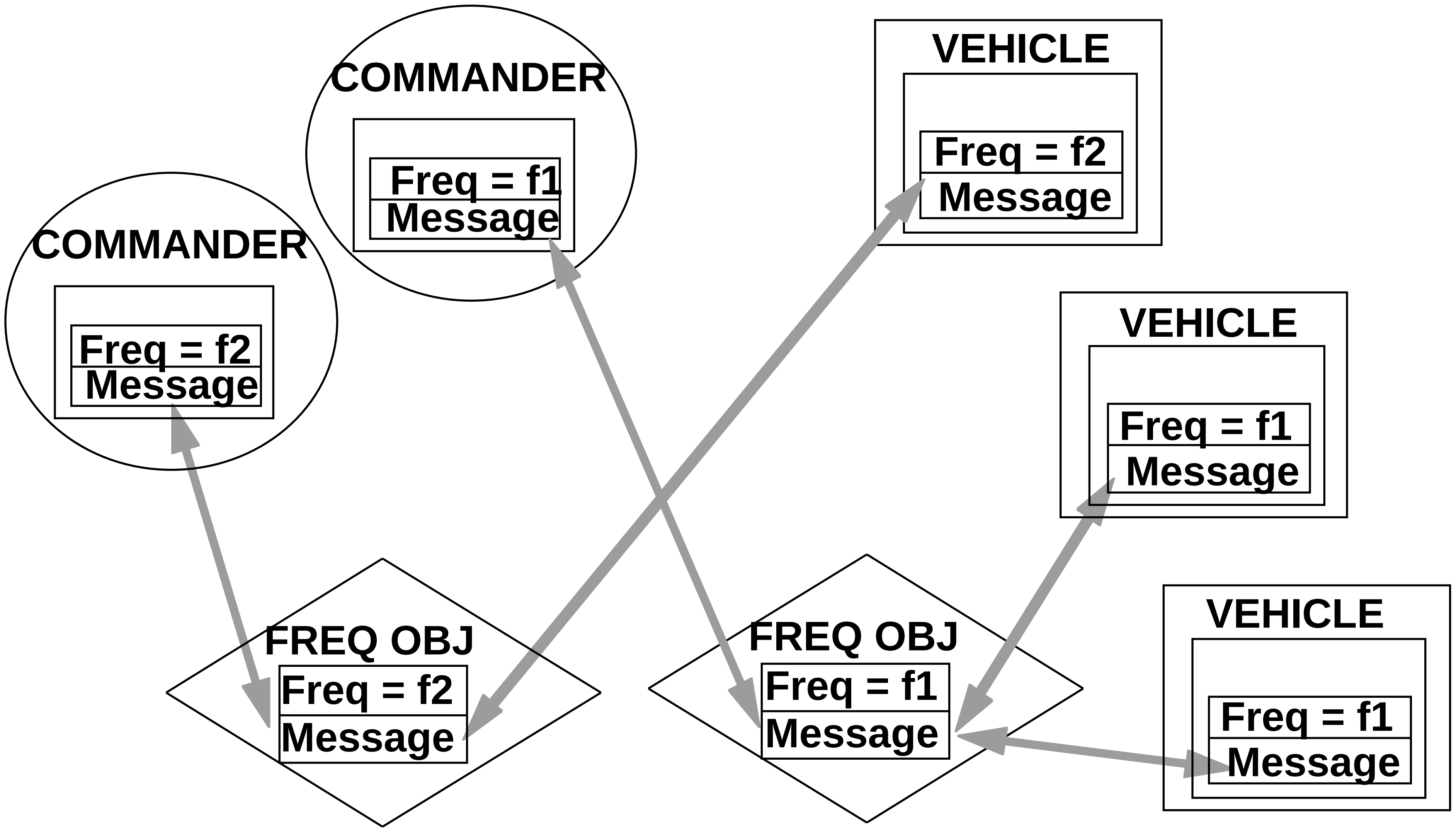
Message Filtering with Subscription by Class



Message Filtering with Subscription by ID



Object Interactions by ID



Software Design Protocols

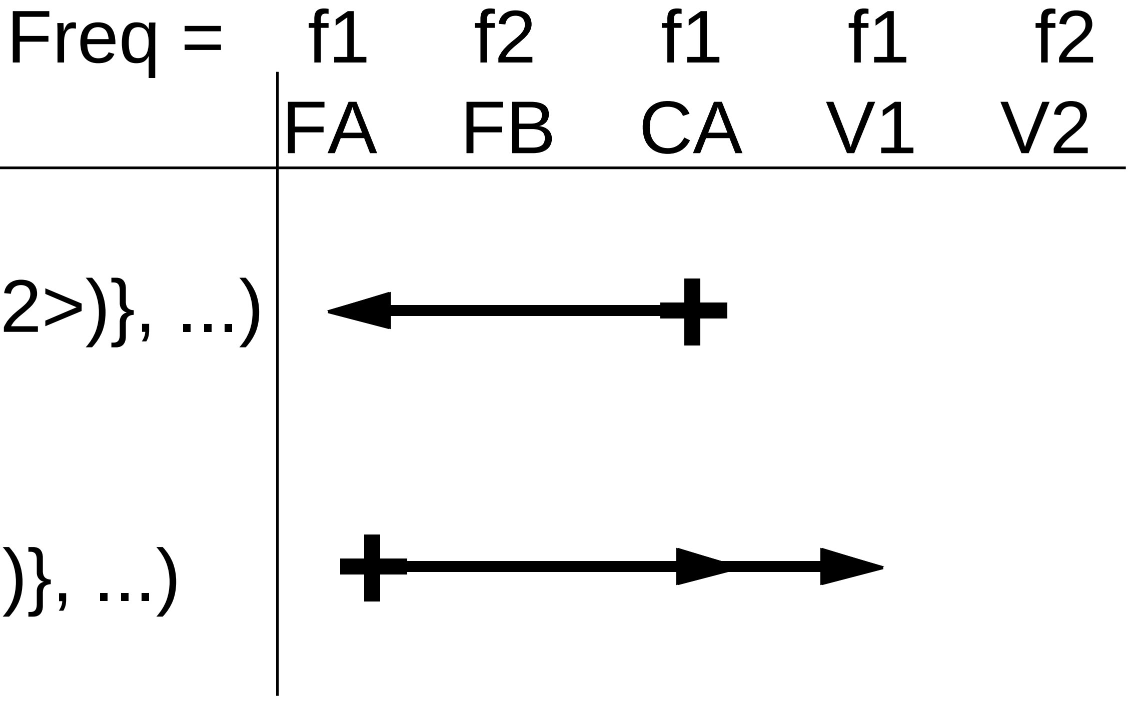
- Message Passing
- Changing Frequencies
- Initialization

Message Passing

- Commander A sends a message (M1) on Frequency 1 (e.g. “Vehicle 1 please move to Position 2”)
- Freq Object A updates both Commander A and Vehicle 1 with message M1

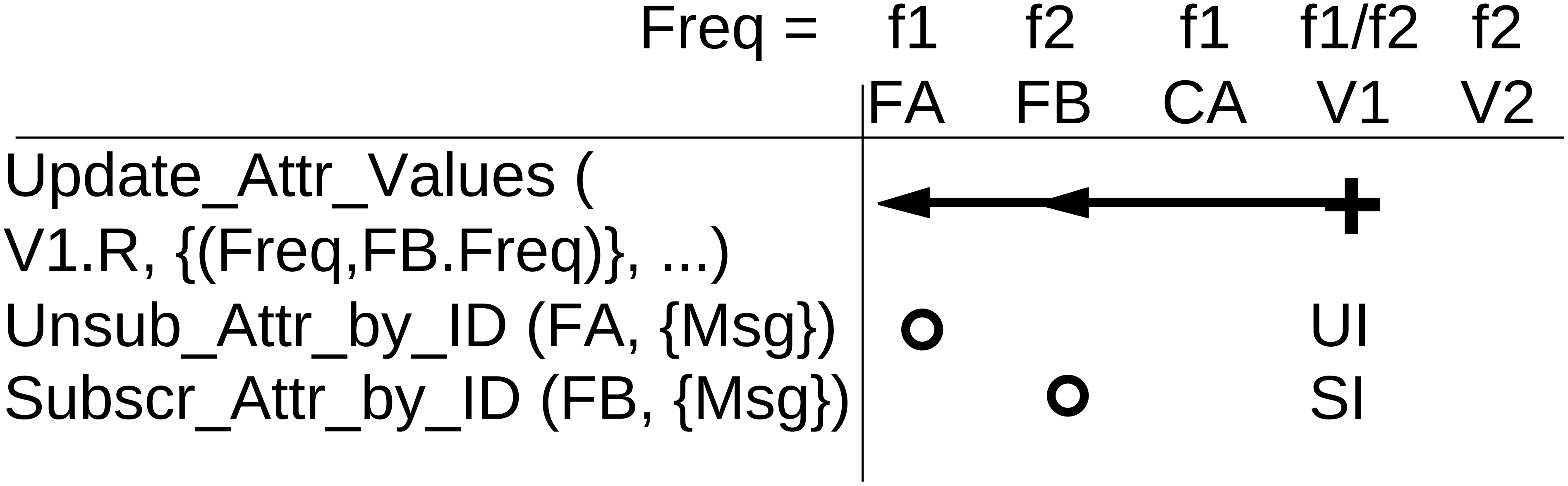
Update_Attr_Value (
CA.R, {(Msg,<M1,V1,P2>)}, ...)

Update_Attr_Value (
FA, {(Msg,<M1,V1,P2>)}, ...)



Changing Frequencies (1)

- Vehicle 1 changes its frequency from f1 to f2 by updating Radio attributes
- Vehicle 1 unsubscribes to Freq Obj A and subscribes to Freq Obj B



Freq = f1 f2 f1 f1/f2 f2
FA FB CA V1 V2
Update_Attr_Values (V1.R, {(Freq,FB.Freq)}, ...)
Unsub_Attr_by_ID (FA, {Msg})
UI
Subscr_Attr_by_ID (FB, {Msg})
SI

Changing Frequencies (2)

- Freq Obj A sees Vehicle 1 has changed frequency, so it unsubscribes to Vehicle 1’s message attribute
- Freq Obj B sees Vehicle 1 has changed to its frequency, so it subscribes to Vehicles 1’s message attribute

Freq =	f1	f2	f1	f1/f2	f2
	FA	FB	CA	V1	V2
Unsub_Attr_by_ID (V1, {Msg})	UI			○	
Subscr_Attr_by_ID (V1, {Msg})		SI		○	

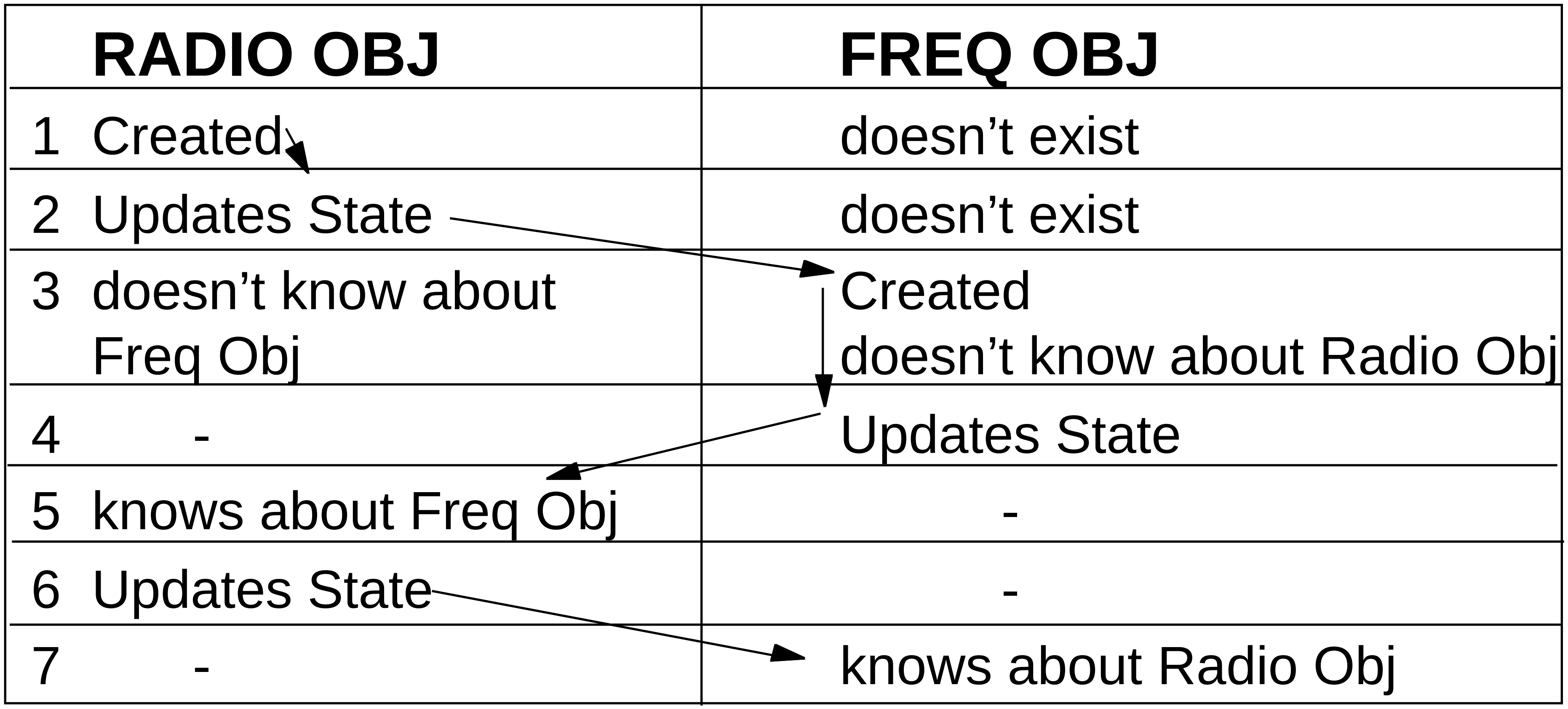
Initialization Overview (Freq Objects Created 1st)

Object Awareness

	FREQ OBJ	RADIO OBJ
1	Created	doesn’t exist
2	Updates State	doesn’t exist
3	doesn’t know about Radio Obj	Created doesn’t know about Freq Obj
4	-	Updates State
5	knows about Radio Obj	-
6	Updates State	-
7	-	knows about Freq Obj

Initialization Overview (Radio Objects Created 1st)

Object Awareness



	RADIO OBJ	FREQ OBJ
1	Created	doesn't exist
2	Updates State	doesn't exist
3	doesn't know about Freq Obj	Created doesn't know about Radio Obj
4	-	Updates State
5	knows about Freq Obj	-
6	Updates State	-
7	-	knows about Radio Obj

Initialization

- Radio and Frequency object attributes are broadcast (using subscription by class) since this is control information
- Message attributes are multicast (using subscription by ID) because there is a lot of this data

Initialization (1)

•All federates that model an object of class Frequency publish its public attributes (Freq and Msg).

•All federates that model an object of class Radio (Commanders and Vehicles) publish its public attributes (Freq and Msg).

	FA	FB	CA	V1	V2
Pub_Obj_Cls (Frequency, {Freq, Msg})	P	P			
Pub_Obj_Cls (Radio, {Freq, Msg})			P	P	P

Initialization (2)

- To learn about existing Freq objects, the radio federates subscribe to attribute Freq of class Frequency
- To learn about existing radio objects, the frequency federates subscribe to attribute Freq of the class Radio
- Each federate informs the RTI about the simulated objects

	FA	FB	CA	V1	V2
Sub_Obj_Cls (Frequency, {Freq})			S	S	S
Sub_Obj_Cls (Radio, {Freq})	S	S			

Initialization (3)

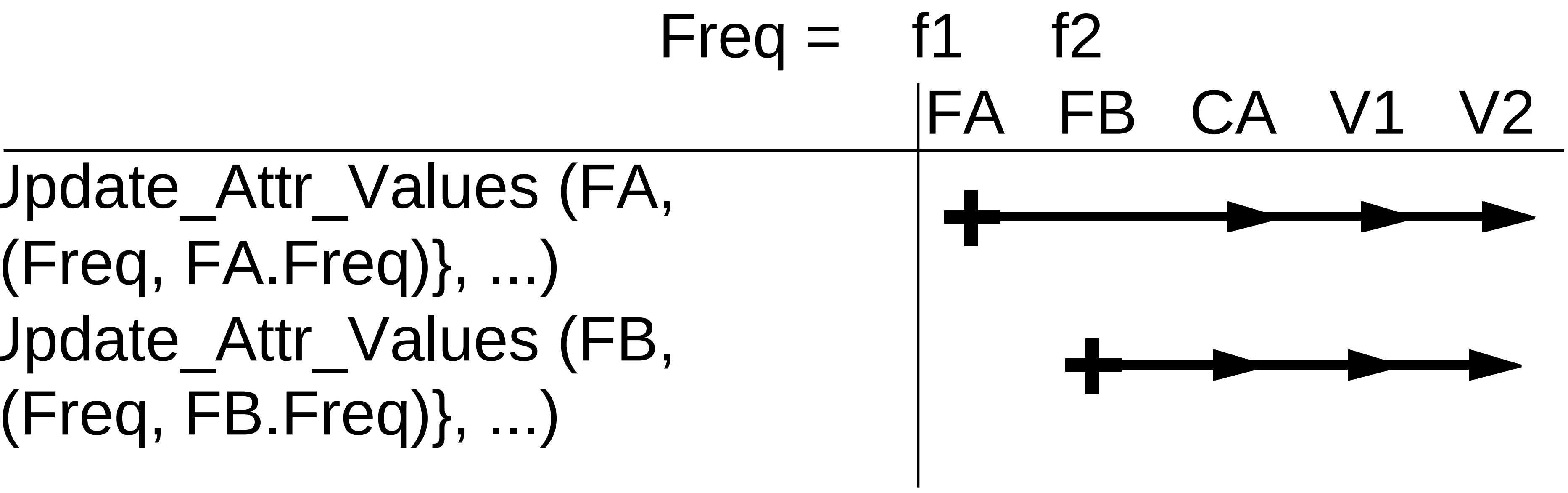
•Each federate informs the RTI about all the simulated objects

	FA	FB	CA	V1	V2
Instatiate_Obj (Frequency, FA)	I				
Instatiate_Obj (Frequency, FB)		I			
Instatiate_Obj (Radio, CA)			I		
Instatiate_Obj (Radio, V1)				I	
Instatiate_Obj (Radio, V2)					I

Initialization (4)

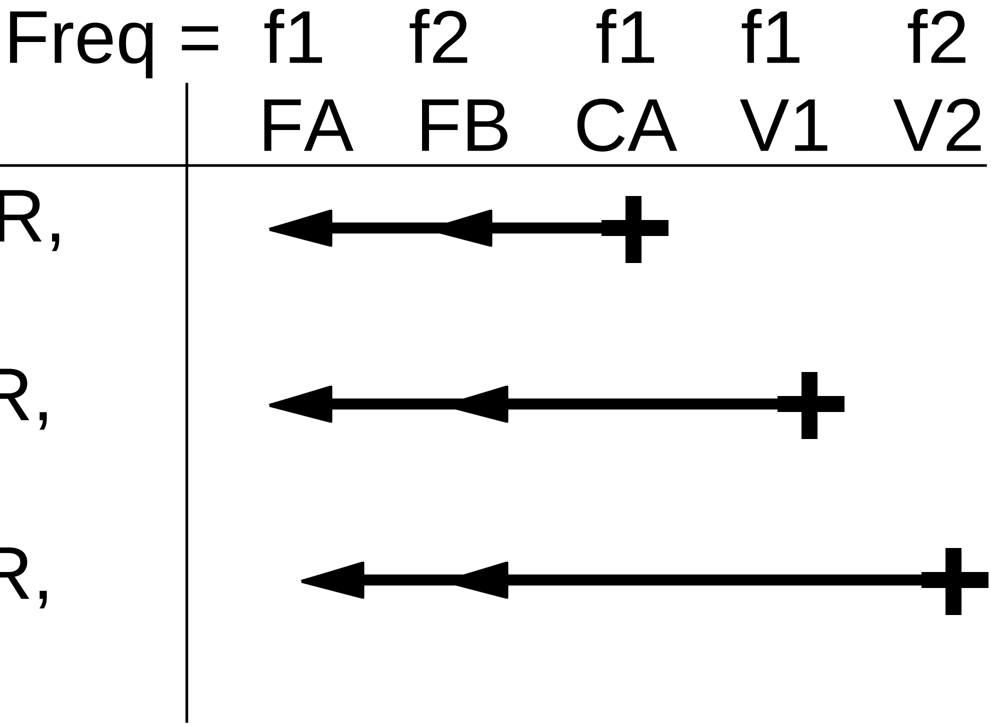
•On creation, each object of class Frequency reports its own Freq.

•All radio objects have already subscribed and therefore receive it.


Freq = f1 f2
FA FB CA V1 V2
Update_Attr_Values (FA, {(Freq, FA.Freq)}, ...)
Update_Attr_Values (FB, {(Freq, FB.Freq)}, ...)

Initialization (5)

- On Creation, each object of class Radio reports its own Freq.
- All the Frequency objects have already subscribed and therefore receive it.



Update_Attr_Values (CA.R, {(Freq, CA.R.Freq)}, ...)

Update_Attr_Values (V1.R, {(Freq, V1.R.Freq)}, ...)

Update_Attr_Values (V2.R, {(Freq, V2.R.Freq)}, ...)

Initialization (6)

•The reflection of the attribute Freq triggers the subscription to the Msg attribute of the corresponding objects.

Freq =	f1	f2	f1	f1	f2
	FA	FB	CA	V1	V2
Subscr_Attr_by_ID (FA, {Msg})	○		SI	SI	
Subscr_Attr_by_ID (FB, {Msg})		○			SI
Subscr_Attr_by_ID (CA.R, {Msg})	SI		○		
Subscr_Attr_by_ID (V1.R, {Msg})	SI			○	
Subscr_Attr_by_ID (V2.R, {Msg})		SI			○

Lessons Learned

- Good:
  - How to use the High Level Architecture
  - How to use the RTI
  - HLA as an application development framework
- Bad:
  - RTI latency is currently too high
  - Subscription by ID is not yet supported
  - Hard to develop to a moving target

Lessons Learned

- HLA
  - Supports improved, more efficient simulation architectures
- SOM/FOM and the Development Process are useful
  - when designing from the ground up

Recommendations

- Testing and Integration should be better supported with:
  - Hooks into the RTI
  - Integration Tools/Libraries

Last Slide